Australian Government
Department of Defence
Defence Science and
Technology Organisation

Using Web Services to Enable the Reuse of DSTO Corporate Data within Microsoft SharePoint: A Case Study

Paul Prekop, Phuong La, Mark Burnett and Chris Chapman

Command and Control Division
Defence Science and Technology Organisation

DSTO-TN-0672

ABSTRACT

This report discusses how Service Oriented Architectures and Web Services were used to make DSTO corporate information available within Microsoft's SharePoint. While the report focuses on exposing of information within SharePoint, the Service Oriented Architecture approach and Web Services technologies are likely to much have wider applicability across the Defence Information Environment.

Published by

Command and Control Division
DSTO Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

Telephone: (08) 8259 5555
Fax: (08) 8259 6567

AR-013-554
December 2005

APPROVED FOR PUBLIC RELEASE

Using Web Services to Enable the Reuse of DSTO Corporate Data within Microsoft SharePoint: A Case Study

Executive Summary

This report discusses an approach to exposing DSTO corporate information within Microsoft SharePoint, using Service Oriented Architectures (SOAs). The SOA approach leads to information systems that are defined in terms of coarse-grained, re-usable business services, rather than low-level business data, components, or systems.

SOAs offer a number of benefits, including that they:

- Encourage and support development of services specifically designed to be re-used. Re-use can reduce development time and reduce the need to duplicate data business logic within applications;
- Encapsulate the complexity of business functions, which can make it easier for developers to build applications; and
- Enable the development of new classes of applications that were too difficult or too expensive to attempt using other approaches.

Within this case study, the SOA approach was implemented using Web Services (WSs). WSs are a set of standards-based technologies that implement SOA concepts.

The objective of this case study was to make DSTO corporate information – accessed via the Automated Research Management System (ARMS) – available within a Microsoft SharePoint portal. ARMS is a prototype information service that collects and fuses a wide collection of existing DSTO corporate information. It exposes information to end-users as a set of dynamically generated web pages and to developers via a set of programmatically accessible WSs.

To expose the corporate information within SharePoint, a collection of Sharepoint Web Parts was developed. Web Parts are custom SharePoint components that can be used to provide custom functionality and access to data sources within SharePoint. The Web Parts developed used the WS interfaces exposed by ARMS to access the corporate information.

This report also discusses the wider applicability of the approaches and technologies used, and discusses several classes of corporate applications that can be supported by the SOA and WS approach.

Contents

1. INTRODUCTION 1

2. MICROSOFT SHAREPOINT 2

3. THE SERVICE ORIENTED ARCHITECTURE (SOA) APPROACH 4
   3.1 SOAs and Web Services 6

4. CORPORATE INFORMATION AND SHAREPOINT 8
   4.1 Exposing Corporate Data and Services 8
   4.2 Accessing Corporate Data in SharePoint 10

5. SOAS AND OTHER CORPORATE APPLICATIONS 12
   5.1 Portal Applications 12
   5.2 Workflow Applications 12
   5.3 Microsoft Office 2003 & New Approaches to Data Access 13

6. CONCLUSIONS 15

7. REFERENCES 17

8. ACKNOWLEDGEMENTS 18

APPENDIX A: ARMS WEB SERVICE 19

APPENDIX B: CURRENT AND EMERGING WEB SERVICE STANDARDS 20

1. Introduction

This report discusses an approach to exposing DSTO corporate information within Microsoft SharePoint, using Service Oriented Architectures (SOAs) and Web Services (WSs). Microsoft SharePoint is a collection of infrastructure technologies that has been used within DSTO in a number of interesting and innovative ways, including task collaboration, supporting research communities, and branch management.

The corporate information exposed within SharePoint is collected via the Automated Research Management System (ARMS). ARMS is a prototype information system aimed at improving information and knowledge sharing and discovery within DSTO. An ARM collects and fuses a wide range of existing DSTO corporate information, and exposes that information to end-users as a set of dynamically generated web pages, via a set of programmatically accessible WSs. The WS component of ARMS was used to access corporate information from within SharePoint.

This report first discusses how SharePoint has been used within DSTO and the types of corporate information needed for the SharePoint sites.

Next this report describes SOAs and WSs in detail. SOAs and WSs were the conceptual and technical approaches used to make the corporate data held in ARMS accessible from within SharePoint.

Section 4 discusses the ARMS WSs and, in particular, how complex applications can be built to include WSs. Structuring applications in such a way as to make it straightforward to include WSs is vital and the methods described in this section are likely to be applicable to most complex information systems. Section 4 also discusses the approaches used to access corporate information from within ARMS and how such information was exposed within SharePoint.

The final section, Section 5, discusses how the SOA and WS approach could be used to enable new classes of useful corporate applications and approaches.

While this report focuses on the use of SOAs and WSs to surface corporate information within SharePoint, many of the techniques and approaches described are equally applicable to other kinds of problems and problem domains.

2. Microsoft SharePoint

Microsoft SharePoint is a collection of infrastructure technologies that can be used to build fully configurable, collaborative, document-centric web sites. The SharePoint infrastructure provides a base set of collaborative features, including: document and picture libraries; discussion forums; security and access control (based on Active Directory); server wide document searching; and user profiles and profile alerting services [1].

The SharePoint infrastructure is fully extendable via software components known as Web Parts. Web Parts are essentially ASP.NET server controls that can be included on SharePoint Web Pages. SharePoint includes a basic set of Web Parts used to support collaboration and document management. It is also possible to develop custom Web Parts to expose new functionality or to provide access to proprietary backend data sources [1].

Within DSTO, SharePoint has been used in a number of ways, including supporting collaborative work within a task, supporting research communities and supporting branch management.

The Theatre Situation Awareness Task (02/212)[1] is an excellent example of using SharePoint to support collaborative work within a task. The task's SharePoint site includes basic information about the task and task contributors as well as a wealth of task documents, including informal and draft task publications, and a library of related and interesting documents and links.

The Human Factors (HF) Hub[2] uses SharePoint to support information sharing and interaction within a DSTO research community. The hub's SharePoint site includes links to past and future hub activities, a document library, links to conferences and events relevant to the community, and a discussion forum.

Joint Command Analysis (JCA) branch uses SharePoint to support branch management and information sharing[3]. The JCA site includes a list of branch contacts, links and branch publications. The site is structured to include sub-sites for each task and group within the branch. Group sites include a list of staff, tasks related to the group, shared documents and group discussions. Task sites include staff working on the task, task deliverables, discussions, shared documents and task financial information.

Common to all these SharePoint sites is the need to include existing corporate information. The Theatre Situation Awareness Task's site, for example, includes descriptive task information, task contributors and task publications. The HF Hub's site includes a list of staff who are working on or interested in human factors. The JCA branch site lists staff in

[1] See: http://c2d-teams.dsto.defence.gov.au/task/02-212/default.aspx
[2] See: http://hubs.dsto.defence.gov.au/sites/HFHub/default.aspx
[3] See: http://c2d-teams.dsto.defence.gov.au/branch/JCA/default.aspx

the branch, the branch's groups, the tasks associated with the branch, task contributors, branch publications and task financial information.

Currently the corporate information held within these sites is manually extracted from existing corporate information systems by the sites' administrators and then manually re-entered into each SharePoint site. This process results in a considerable duplication of effort, with the same corporate data held in several locations as well as being keyed in several times. Also, since there is no overarching control of the data, the data is held in different locations by different systems and rapidly becomes out-of-date.

The goal of this case study was to explore ways of exposing existing corporate information within the various SharePoint sites, without having to hold local copies of data and without needing to re-key data into the SharePoint sites. The next few sections explore both a conceptual approach to addressing this problem as well as the specific technology used. While this report focuses on the issues relating to accessing corporate information from within SharePoint, the issues discussed are not unique to SharePoint; sharing core data between different corporate applications is a ubiquitous problem within the Information Systems domain.

3. The Service Oriented Architecture (SOA) Approach

Service Oriented Architecture is an approach to building information systems that are defined in terms of coarse-grained, re-usable business services, rather than low-level business data, components, or systems. Like object oriented design and development before it, the SOA approach encourages the designing and building of information systems by modelling and representing the functions or services of the business, rather than low-level programming constructs [2].

In the SOA approach, common, atomic, business functions and processes are exposed via an appropriate middleware technology (one commonly used middleware technology is Web Services, discussed in the next section). End-user applications can then combine the business functions exposed as services to provide end-users with the functions needed to support a complete business process or activity [2].

The biggest conceptual challenge of the SOA approach is that it requires a much fuller and deeper understanding of the organisation's information environment. Rather than thinking about individual information systems as discrete, stove-piped applications, the SOA approach encourages thinking about information systems as contributors to an organisation-wide information environment or architecture, that consists of common organisational services (in terms of data and functions). It also encourages thinking of end-user focused functionality as the aggregation of common services to support specific business processes or activities.

For example, within DSTO, two important groups of common business functions are financial information and management and task information and management. Each of these groups might expose a set of re-usable business services related to financial and task information and management. These services may, for example, include: a *Get Task Descriptive Summary* service, or *Debit Account (WA) Number with Amount for Line Item* service. This hypothetical architecture is shown in Figure 1.

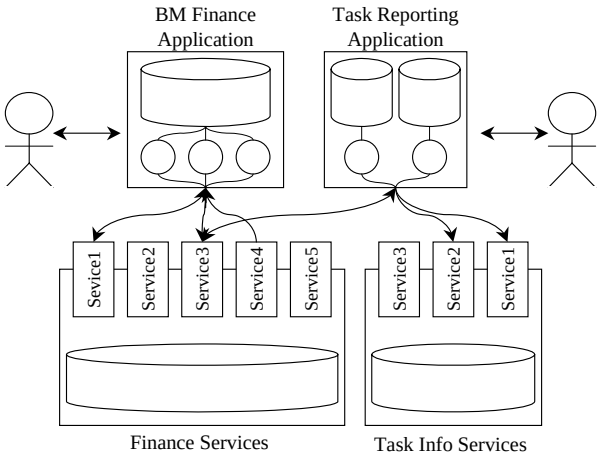

Figure 1: Hypothetical SOA Architecture

As shown in Figure 1, individual applications, for example a *Business Managers (BM) Finance Application*, or a *Task Reporting* application would use the exposed financial and tasks management services, together with local data and local functionality, to provide end-users with a complete business application.

One of the biggest advantages of the SOA approach is the encapsulation of complex business functions, business logic and business data behind well-defined, coarse-grained, business-focused interfaces. Encapsulation can help encourage the development of services specifically designed to be re-used. For example a *Debit Account (WA) Number with Amount for Line Item* service, that may form part of a financial management group of functions, could be re-used by many different applications, for example an *Order Processing Work Flow Application* and a *Contractor Payment System*. Building end-user applications from a collection of existing services can help reduce the time and effort required to develop new applications.

Since encapsulation uses well-defined interfaces to decouple service from applications that use them, it becomes easier to manage change and complexity. Applications make use of services via the service's well-defined interface. As a result, applications make no assumptions about the actual implementation of the service. If the implementation of the service were to change, applications that use the service are generally well insulated from the changes and are less likely to require redevelopment.

The interfaces used to expose the services also hide the complexity of the implementation from developers because the interfaces are drawn from the business domain, meaning developers can work with business-focused interfaces, rather than dealing with abstract programming constructs.

The SOA approach can also enable development of new classes of applications that were too difficult or too expensive to attempt when working with more tightly coupled architectures. As discussed in Section 5, the SOA approach can make it easier to develop portals and workflow applications because the business functions these applications depend on can be separated from the portal or workflow. This makes it easier to create flexible workflow or portal applications that draw business functions from the well-defined service interfaces.

While the SOA approach can offer considerable time and cost savings over the medium and long term, as well as providing greater flexibility and the ability to develop new classes of applications, the SOA approach can require considerably more initial effort than other approaches. Developing services with re-use in mind can make them more expensive to develop because they need to be built in ways that makes them generalisable across a variety of applications and uses. Also, the SOA approach depends on a deeper understanding of the underlying business models the services and applications need to support. These models can be time consuming and expensive to capture and maintain.

Since the SOA approach assumes that applications are built from a collection of services that may be distributed anywhere on the network, achieving appropriate application responsiveness could require reliable, high bandwidth networks. In some Defence contexts, this requirement may be difficult to meet.

In environments with a large collection of legacy systems, converting existing tightly coupled legacy systems to a collection of services can require considerable engineering effort, especially if the existing systems are single or two tier architectures with tightly coupled interface, business logic and database layers. This adds to the cost and effort of adopting an SOA approach.

3.1 SOAs and Web Services

Web Services (WSs) are a set of standards-based technologies that supports the concept of Service Oriented Architectures (SOAs) discussed in the previous section. While SOA concepts are not specifically tied to WS technology, the concepts of SOAs and WSs have evolved together, and as a result WSs provide all of the functions needed to support a SOA system development approach.

WSs are an open, standards-based framework that consists of several major components, each providing part of the functionality needed to implement SOA concepts.

XML (Extensible Markup Language) is core to all WS components. XML is used as a common method for describing data and data structures that are passed to and returned from WSs. Using XML to describe data makes it possible to disentangle data from the underlying application specific structures used to store it. This makes it possible to describe data in more widely accessible, application independent ways.

For WSs to be able to exchange useful functionality, some kind of communications protocol is needed. The communications protocol commonly used to expose WSs is SOAP (Simple Object Access Protocol). SOAP is a standards based protocol that describes the format of messages passed to and returned from WSs.

As a communications protocol, SOAP supports two major access paradigms – traditional RPCs (Remote Procedure Calls) and an interesting Document-Centric approach. In the Document-Centric approach the WSs operate on complete XML documents rather than just simple parameters as in RPC approach. Since Document-Centric WSs operate on full XML documents, state and other transaction dependent information can be passed to the services as part of the XML document during execution. This can result in development of very flexible and generic WSs, and can remove much of the complexity needed to manage state information.

SOAP is generally bound to HTTP (Hypertext Transfer Protocol) as the underlying communication transport; however it can be bound to other transport layers, for example

MSMQ, MQ Series or even SMTP. The HTTP binding is the most useful, because almost all organisations have a widely deployed HTTP infrastructure.

While SOAP describes the underlying messaging and communications protocols needed to use a WS, WSDL (Web Services Description Language) formally defines the WS's interface. A WS's WSDL description is an XML file that describes everything a developer or an application needs to know to be able to programmatically access the WS, including descriptions of the parameters, their structure and type, the return types and structures, the location of the WS, and the communications bindings needed to access the WS.

The final core WS technology component is service registration. A common method used to support service registration is UDDI (Universal Description, Discovery and Integration). UDDI is a standard for publishing and discovering the existence of services on a network. External UDDI registries[4] allow organisations to publish and access services available in the public domain. UDDI can also be used internally within an organisation to publish and access services. This can be very useful within large organisations, or organisations with a diverse and distributed developer community.

In addition to the core WS components described in this section, Appendix B: describes several additional standards built on top of the basic WS infrastructure, including transaction support, security and authentication.

[4] For example, see: http://uddi.ibm.com or http://uddi.microsoft.com.

4. Corporate Information and SharePoint

As discussed in Section 2, the goal of this work was to expose existing corporate information within SharePoint sites without having to hold local copies of data or to re-key data into SharePoint. The approach taken consists of two key parts – a set of Web Services (WSs) that make the existing corporate information available (see Section 4.1), and a set of SharePoint Web Parts that use the WSs and surface corporate information within SharePoint (see Section 4.2).

4.1 Exposing Corporate Data and Services

The Automated Research Management System (ARMS) [3] is a prototype information system built to explore ways of improving information and knowledge sharing and discovery with DSTO. ARMS accesses and fuses a wide collection of DSTO corporate information, including staff/task, publications and other task outcome information, as well as organisational structure information. ARMS exposes this information to end-users as a set of richly contextualised, dynamically generated web pages, as well as via powerful multi-dimensional search and configurable browse functions. ARMS also includes a WS interface that provides query services to access data held by ARMS. Figure 2 shows the high-level conceptual ARMS architecture.

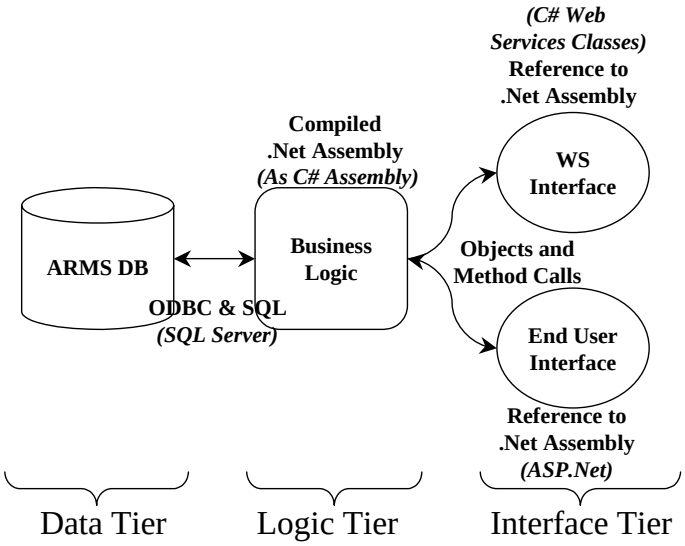

Figure 2: Conceptual ARMS Architecture

ARMS uses a three tier architecture consisting of a Data Tier, Business Logic and an Interface Tier. The key advantage of this architectural approach is that business logic is encapsulated, allowing it to be decoupled from the data and interface tiers. This decoupling means the same business logic can be re-used by different interfaces. ARMS provides two interfaces into the business logic – a programmatic WS interface and the end-user Web based interface. Both interfaces use the same business logic to provide access to

the ARMS data and functions; they simply act as wrappers over the business logic tier. This approach makes the interface tiers simpler and decouples them from the details of how ARMS data is stored and manipulated.

The WS interface exposes a simple set of Query-by-ID methods for the following business objects: staff, output, tasks and unit. The WS interface also exposes a search service and several simple data conversion services (See Appendix A for a description of the full ARMS WS interface).

All of the WS methods return XML documents that describe a business object or, more commonly, collections of business objects. Figure 3[5], for example, shows part of the XML structure for an output object.

```
<?xml version="1.0" encoding="utf-8"?>
<Output ...>
    <Title>Deaf, Dumb And Stupid: Harnessing Evolution To Create Succesful And Adaptiv
    ...
    <ContributingTasks>
        <Task>
            <ID>01/116</ID>
            <TaskName>ER&amp;D IN PERVASIVE COMPUTING</TaskName>
            ...
            <TaskManager>
                <Staff>
                    <Title>Dr</Title>
                    <Firstname>Mark</Firstname>
                    <Surname>Burnett</Surname>
                    ...
                    <Unit>
                        <Fullname>Information Systems Group</Fullname>
                        ...
                    </Unit>
                    <Site>
                        <Name>DSTO Fern Hill Park</Name>
                        ...
                    </Site>
                </Staff>
            </TaskManager>
            <LeadUnit>
                <Unit>
                    <Fullname>Information Systems Group</Fullname>
                    ...
                </Unit>
            </LeadUnit>
        </Task>
    </ContributingTasks>
    <Authors>
        <Staff>
            <Title>Mr</Title>
            <Firstname>Tim</Firstname>
            <Surname>Smith</Surname>
            ...
```

Figure 3: Output Object's Top Level Structure

An important characteristic of the objects returned by the ARMS WS is that they are de-normalised and complete. As shown in Figure 3, the output object includes copies of all the other business objects related to the particular output object: the staff objects related to this output via the authorship relationship, the task object related to the output via the task relationship, and so on. Since the XML documents are de-normalised and complete,

[5] Some of the actual data has been removed to make the structure explicit

all of the complexity involved in actually querying the database and executing the business logic to create the complete business objects is encapsulated within the WS.

4.2 Accessing Corporate Data in SharePoint

As discussed in Section 2, SharePoint can be fully extended via Web Parts. Web Parts are a type of ASP.NET server control that can be included on SharePoint web pages and can be used to provide any kind of custom functionality or access to custom data sources. To expose DSTO corporate information within SharePoint, six Web Parts were developed:

- Task Information (ArmsTaskInfo) – shows basic task information. Queried by Task Number;
- Task Contributions (ArmsTaskContributors) – shows a list of staff that have contributed to a particular task. Queried by Task Number;
- Task Publications (ArmsTaskPublications) – shows a list of publications attributed to a particular task. Queried by Task Number;
- Unit Staff (ArmsUnitStaff) – shows a list of staff associated with a particular unit. Queried by ARMS Unit Identifier. The selection of staff is recursive over child units;
- Unit Tasks (ArmsUnitTasks) – shows a list of tasks associated with a particular unit. Queried by ARMS Unit Identifier. The selection of tasks is recursive over child units; and
- Unit Publication (ArmsUnitPublications) – shows a list of publications attributed to a particular unit. Queried by ARMS Unit Identifier. The selection of publications is recursive over child units.

All of the Web Parts were developed in C# and all access corporate information via the ARMS Web Services (WSs) interface. Since the Web Parts access corporate data via the ARMS WSs, they contain no business logic and do not depend on the underlying database structures or databases queries to access or manipulate ARMS data. Since all the complexity of querying the ARMS database is encapsulated behind the WS interface, the Web Parts are very simple; they only need to maintain a simple local state, query the well-defined ARMS WS interface and display the resulting data.

As part of their default functionality, Web Parts contain a set of enduring properties that define their behaviour, their look and feel, and so on. In addition to the standard set of properties, each of the six Web Parts developed as part of this case study also included a custom property used to hold the query parameter to be passed to the ARMS WS. In most cases, the query parameter is a simple object identifier. The query parameter is set when the Web Part is first added to the SharePoint page.

As shown in Figure 4, once added to the SharePoint page, the Web Part is executed whenever a user loads this page. On execution, the Web Part simply calls the appropriate ARMS WS, passing it the stored query parameter. The ARMS WS returns the required information as an XML document. The Web Part then renders the XML as HTML

dynamically via Extensible Stylesheet Language for Transformation (XSLT). XSLT is described in more detail in Appendix B.2.

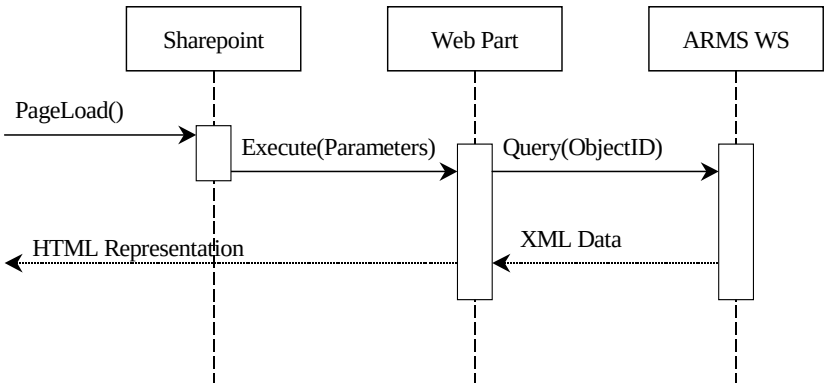

Figure 4: Web Part Sequence Diagram

As shown in Figure 4, the Web Parts developed to access and expose corporate information within SharePoint are very simple; they simply access ARMS data and use it in ways that make sense to the Web Part. They hold no ARMS business logic and are not coupled in any tight way to the underlying ARMS data tier.

5. SOAs and other Corporate Applications

As well as providing a means of exposing corporate data within SharePoint, the Service Oriented Architecture (SOA) and Web Service (WS) approach described in this report can also be use to support other classes of corporate applications. In many cases the kinds of applications supported by SOAs and WSs would be very difficult or expensive to develop using traditional approaches.

5.1 Portal Applications

A key class of applications that can be supported by the SOA and WS approach is Portals. Portals pull together information from a variety of information sources into the one location. All of the SharePoint sites discussed in Section 2 can be seen as examples of end-user developed Portal applications.

As shown by this case study, accessing corporate data and services exposed via WSs means that the resulting Portal applications are very simple and lightweight because they don't need to directly hold or manage corporate data; all of the complexity and business logic involved in manipulating corporate data is encapsulated within the WS. In most cases all the Portal applications need to do is to query the WSs and display the resulting data in ways that make sense within the Portal.

Also, since the Portal application queries the WS (and hence the underlying corporate data source) at run time, the Portal application can access the most up-to-date information created by the most up-to-date set of business logic.

Many portal development frameworks, for example Microsoft SharePoint and IBM WebSphere, can directly or indirectly provide methods to access data via WSs.

5.2 Workflow Applications

Another key class of applications that the SOA approach enables is workflow applications. Workflow applications generally consist of two major components: sets of functions that need to be performed by corporate applications in response to workflow events and workflow logic that orchestrates workflow events. The SOA approach naturally supports sophisticated workflow because it encourages exposing of core business functionality and objects through well-defined interfaces. These interfaces can then be used by the workflow logic to access and modify corporate data.

An example SOA and WS based workflow architecture is shown in Figure 5. In this example, three corporate applications – a Finance System, a Task Management Information System (TMIS) and a Staff System – all expose their core business services via a set of WS interfaces (shown as WS1, WS2, etc in the figure). A workflow orchestration service, such as Microsoft's BizTalk Server, can then use these exposed WS interfaces to

thread a workflow process (shown in Figure 5 as the link threading between the three systems) between the various systems, using exposed services to access and update corporate data as the workflow progresses.

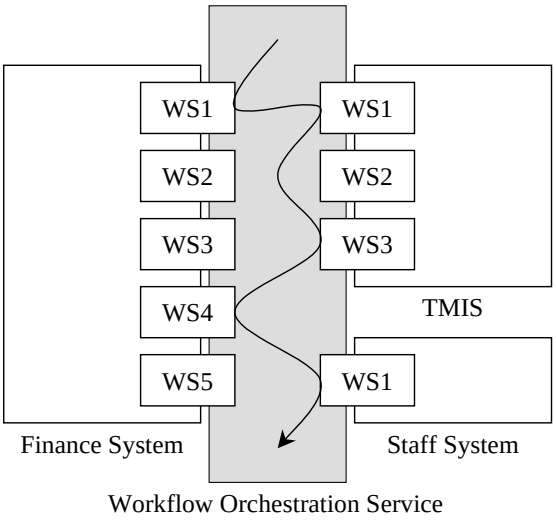

Figure 5: Example SOA and WS based workflow architecture

A key advantage of this workflow architecture is that the workflow logic is external to the underlying services used to execute the workflow (i.e. the WS interfaces). This simplifies the overall design of the workflow application and allows any number of different workflows to use the same set of exposed WS interfaces in completely different workflows.

Separating the workflow logic from the underlying services that provide the functionality into a separate workflow orchestration means that the workflow logic can be changed independently of the underlying services that support it. New workflows can be added and existing workflows can be modified without the need to change any of the underlying information systems.

5.3 Microsoft Office 2003 & New Approaches to Data Access

The latest version of Microsoft Office, Office 2003, takes advantage of Web Service (WS) technologies with the introduction of two interesting features: Research Services[6] and Microsoft Office Information Bridge Framework[7] (MOIBF).

Research Services is a standard feature of Office that provides a mechanism for accessing WSs without the user needing to leave an Office application. Data sources that are exposed as WSs, such as Encarta, HighBeam's eLibrary, Microsoft SharePoint,and MSN Search, or custom data sources (for example the ARMS WS interface described in Section 4.1) can be queried via a standard search interface embedded within all Office applications. The

[6] See: http://msdn.microsoft.com/office/understanding/research/default.aspx
[7] See: http://msdn.microsoft.com/office/understanding/ibframework/default.aspx

results of searches can be inserted into a current Office document, copied to the clipboard, linked to a URL, or become the seed of a new query submitted to the services.

The second key feature is the Microsoft Office Information Bridge Framework (MOIBF). MOIBF builds on the functionality provided by Research Services and enables an Office application's Task Pane to be customised to provide flexible access to any number of WS exposed data sources at once. This allows a wide collection of data sources to be accessed, combined and presented to the user directly within an Office application.

MOIFB also supports the concept of Smart Tags. Within Office documents, particular types of strings, for example Task Numbers or other identifiers, can be defined as Smart Tags. Whenever a Smart Tag is encountered within an Office document, Office applications can be set-up to perform custom actions. For example, when entering a Task Number (a unique DSTO identifier assigned to each task) into a Word document, Word could query a task repository via its exposed WS to retrieve the task's information and display it within Word's Task Pane.

6. Conclusions

This report discusses an approach to exposing DSTO corporate information within Microsoft SharePoint. To make corporate information accessible from within SharePoint, a Service Oriented Architecture (SOA) approach was used. Within this case study the SOA approach was implemented using Web Services (WSs).

DSTO corporate information exposed within SharePoint was accessed via the Automated Research Management System (ARMS). As well as exposing information as a set of dynamically generated web pages, ARMS also exposes information via a set of programmatically accessible WSs. To access corporate information from within ARMS, a collection of SharePoint Web Parts was developed. These Web Parts use WS interfaces exposed by ARMS to access corporate information.

Use of SOAs and WSs has several key advantages, including:

- The encapsulation of complex business functions and data behind well-defined, coarse-grained, business focused interfaces can help encourage development of services specifically designed to be re-used. Building end-user applications from a collection of existing services can help reduce the time and effort required to develop new applications, and allows core business logic to be re-used by many different applications;
- The interfaces used to expose services also hide the complexity of the implementation from developers, meaning developers can work with business-focused interfaces, rather than dealing with abstract programming constructs;
- The services can be used in contexts not known as design time. The use of ARMS services in SharePoint described in this report is an example here; and
- The SOA approach can also enable development of new classes of applications that were too difficult or too expensive to attempt when working with more tightly coupled architectures.

While this case study has focused on accessing information within ARMS and exposing it within SharePoint, the approaches and technologies described in this report are likely to have much wider applicability, especially in development of more general purpose portal and work flow applications, and use of innovative new approaches to accessing corporate data.

In addition, the SOA approach (and the case study described here) provides a powerful and practical way of addressing the issue of sharing core data between different corporate applications.

However, developing services with re-use in mind can add to the initial development cost of applications, and redeveloping existing legacy systems to expose their functionality as a collection of loosely coupled services can require considerable effort. In addition, the SOA

approach depends on a deeper understanding of the underlying business models the services and applications need to support. These models can be time consuming and expensive to capture and maintain.

But by far the biggest challenge of the SOA approach is the change in thinking required. Rather than thinking about individual information systems as discrete, stove-piped applications, the SOA approach requires thinking about information systems as contributors to an organisation-wide information environment or architecture, consisting of common organisational services (in terms of data and functions), as well as thinking of end-user focused functionality as the aggregation of common services to support specific business processes or activities.

7. References

[1] L. Langfeld, C. Spence, and M. Noel, *Microsoft Sharepoint 2003*. Indianapolis: SAMS, 2004.

[2] D. Kaye, *Loosely Coupled: The Missing Pieces of Web Services*. Marin County: RDS Press, 2003.

[3] P. Prekop, M. Burnett, and C. Chapman, "The Prototype Automated Research Management System (ARMS)," Defence Science and Technology Organisation (DSTO), Edinburgh, South Australia, Technical Note DSTO-TN-0540, February 2004.

8. Acknowledgements

We thank Holger Kohler and Mark Nelson for all their help with SharePoint.

Appendix A: ARMS Web Service

A.1. Staff Object

- GetStaff (StaffID) Returns: A complete Staff Record.
- GetTask (StaffID) Returns: Tasks a staff member is associated with.
- GetTaskStatus (StaffID, TaskStatusCode) Returns: Tasks a staff member is associated with.
- GetOutput (StaffID) Returns: Outputs a staff member is an author or co-author of.
- LookUpAGS (AGS Number) Returns: Internal ARMS StaffID.

A.2. Output

- GetOutput (OutputID) Returns: A complete output record.

A.3. Task

- GetStaff (TaskID) Returns: Staff associated with the task.
- GetTask (TaskID) Returns: A complete task record.
- GetOutput (TaskID) Returns: Outputs associated with the task.

A.4. Unit

- GetStaff (UnitID) Returns: The staff associated with the unit.
- GetUnit (UNitID) Returns: A complete unit record.
- GetTask (UnitID) Returns: The tasks associated with the unit.
- GetTaskStatus (UnitID, TaskStatusCode) Returns: The tasks associated with the unit.
- GetOutput (UnitID) Returns: Computer Code output record; Data output record; Draft Publication output record; Other output record; Task Note output record; Task Presentation output record; DSTO Report output record.
- LookUpUnitID (UnitAbbreviation) Returns: Internal ARMS UnitID.

A.5. Search

- Query (Query, ReturnObjects) Returns a search result.

Appendix B: Current and Emerging Web Service Standards

B.1. XML

Extensible Markup Language (XML) is a mark-up language much like HTML that conforms to the SGML (Structured Generalised Markup Language) model. It was designed to describe the content in terms of what data is being described. It allows authors to define their own tags that relate to the document content.

XML uses a similar tag structure to HTML, however HTML defines how elements are displayed, XML defines what those elements contain. While HTML uses predefined tags, XML allows tags to be defined by the developer of the page.

See: http://www.w3.org/XML/.

B.2. XSLT

Extensible Stylesheet Language for Transformations (XSLT) is a stylesheet language for XML. XSLT is a standard way to describe how to transform the structure of an XML document into an XML document with a different structure. XSLT is a recommendation of the World Wide Web Consortium (W3C).

See: http://www.w3.org/TR/xslt.

B.3. SOAP

Simple Object Access Protocol (SOAP) is an XML-based protocol for accessing a Web Service. It describes the contents of a message and how to process it, and offers a transport binding for exchanging messages.

See: http://www.w3.org/TR/soap/.

B.4. WSDL

Web Services Description Language (WSDL) is an XML-based language for specifying the interface to a Web Service that describes what a web service can do, where it resides and how to invoke it. It provides a way for service providers to describe the basic format of web service requests over different protocols.

See: http://www.w3.org/TR/wsdl.

B.5. UDDI

Universal Description, Discovery and Integration (UDDI) is the naming service standard for XML Web services. It provides a mechanism for clients to dynamically find other web services.

See: http://www.uddi.org.

B.6. BPEL

Business Process Execution Language for Web Service (BPEL4WS), commonly called BPEL, is a Web Service orchestration standard. It was jointly developed by BEA Systems, IBM and Microsoft. BPEL combines and replaces Microsoft's XLANG specification and IBM's Web Services Flow Language (WSFL). It is an XML-based language that is used to coordinate web services across a single business process.

Microsoft's XML Language (XLANG) is the orchestration language used by Microsoft's BizTalk server.

IBM's Web Services Flow Language (WSFL) is an XML-based language that provides the ability to describe workflows and to call Web services to orchestrate workflows.

BPEL enables the top-down realisation of Service Oriented Architectures (SOAs) through composition, orchestration and coordination of Web Services. BPEL allows developers to specify business processes and how they relate to Web Services. This includes specifying how a business process makes use of Web Services to achieve its goal, as well as specifying Web Services that are provided by a business process.

See: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.

B.7. WS-Security

Web Services Security (WS-Security) is a standard released by OASIS in March 2004. It describes security-related enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality and single message authentication.

See: http://www-106.ibm.com/developerworks/webservices/library/ws-secure/.

B.8. WS-Trust

Web Services Trust (WS-Trust) defines extensions that build on WS-Security to provide a framework for requesting and issuing security tokens, and to broker trust relationships.

See: http://www-106.ibm.com/developerworks/library/specification/ws-trust/.

Distribution List

Using Web Services to Enable the Reuse of DSTO Corporate Data within Microsoft SharePoint: A Case Study:

Paul Prekop, Phuong La, Mark Burnett and Chris Chapman

Task Sponsor

Deputy Chief Defence Scientist Policy	1 Printed

S&T Program

Chief Defence Scientist	1
AS Science Corporate Management	1
Director General Science Policy Development	1
Counsellor Defence Science, London	Doc Data Sheet
Counsellor Defence Science, Washington	Doc Data Sheet
Scientific Adviser to MRDC, Thailand	Doc Data Sheet
Scientific Adviser Joint	1
Navy Scientific Adviser	Doc Data Sht & Dist List
Scientific Adviser – Army	Doc Data Sht & Dist List
Force Scientific Adviser	Doc Data Sht & Dist List
Scientific Adviser to the DMO	1

Information Sciences Laboratory

Chief of Command and Control Division	Doc Data Sht & Dist List
Research Leader, Information Enterprises Branch	Doc Data Sht & Dist List
Head Information Systems Group	1
Paul Prekop	2 Printed
Phuong La	2 Printed
Mark Burnett	2 Printed
Chris Chapman	2 Printed

DSTO Library and Archives

Library Edinburgh	1 Printed
Defence Archives	1 Printed
Library Canberra	1 Printed

Defence Science and Technology Organisation

Manager, Library and Information Systems (MLIS)	1
Manager, Knowledge Management Projects, DSTO Research Library	1
Manager, Corporate Business Solutions	1 Printed

Manager, Management Information Systems Section, Corporate Business Solutions	1 Printed
Capability Development Group	
Director General Maritime Development	Doc Data Sheet
Director General Capability and Plans	Doc Data Sheet
Assistant Secretary Investment Analysis	Doc Data Sheet
Director Capability Plans and Programming	Doc Data Sheet
Chief Information Officer Group	
Director General Australian Defence Simulation Office	Doc Data Sheet
AS Information Strategy and Futures	Doc Data Sheet
Director General Information Services	Doc Data Sheet
Strategy Group	
Director General Military Strategy	Doc Data Sheet
Assistant Secretary Governance and Counter-Proliferation	Doc Data Sheet
Navy	
Maritime Operational Analysis Centre, Building 89/90 Garden Island Sydney NSW Deputy Director (Operations) Deputy Director (Analysis)	Doc Data Sht & Dist List
Director General Navy Capability, Performance and Plans, Navy Headquarters	Doc Data Sheet
Director General Navy Strategic Policy and Futures, Navy Headquarters	Doc Data Sheet
Air Force	
SO (Science) - Headquarters Air Combat Group, RAAF Base, Williamtown NSW 2314	Doc Data Sht & Exec Summary
Army	
ABCA National Standardisation Officer Land Warfare Development Sector, Puckapunyal	e-mailed Doc Data Sheet
SO (Science) - Land Headquarters (LHQ), Victoria Barracks NSW	Doc Data Sht & Exec Summary
SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L), Enoggera QLD	Doc Data Sheet
Joint Operations Command	
Director General Joint Operations	Doc Data Sheet
Chief of Staff Headquarters Joint Operations Command	Doc Data Sheet
Commandant ADF Warfare Centre	Doc Data Sheet
Director General Strategic Logistics	Doc Data Sheet
COS Australian Defence College	Doc Data Sheet
Information Manager, HQJOC Potts Point, Sydney	1 Printed
Intelligence and Security Group	
AS Concepts, Capability and Resources	1

DGSTA , Defence Intelligence Organisation	1 Printed
Manager, Information Centre, Defence Intelligence Organisation	1
Director Advanced Capabilities	Doc Data Sheet

Defence Materiel Organisation

Deputy CEO	Doc Data Sheet
Head Aerospace Systems Division	Doc Data Sheet
Head Maritime Systems Division	Doc Data Sheet
Program Manager Air Warfare Destroyer	Doc Data Sheet
CDR Joint Logistics Command	
Guided Weapon & Explosive Ordnance Branch (GWEO)	Doc Data Sheet
Program Manager JCSE	4 Printed

OTHER ORGANISATIONS

National Library of Australia	1
NASA (Canberra)	1

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy

Library	1
Head of Aerospace and Mechanical Engineering	1
Hargrave Library, Monash University	Doc Data Sheet

OUTSIDE AUSTRALIA

INTERNATIONAL DEFENCE INFORMATION CENTRES

US Defense Technical Information Center	1
UK Dstl Knowledge Services	1
Canada Defence Research Directorate R&D Knowledge & Information Management (DRDKIM)	1
NZ Defence Information Centre	1

ABSTRACTING AND INFORMATION ORGANISATIONS

Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts, US	1
Documents Librarian, The Center for Research Libraries, US	1

SPARES	5 Printed

Total number of copies: 48 Printed: 25 PDF: 23

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Using Web Services to Enable the Reuse of DSTO Corporate Data within Microsoft SharePoint: A Case Study

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

Paul Prekop, Phuong La, Mark Burnett and Chris Chapman

5. CORPORATE AUTHOR

Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TN-0672	AR-013-554	Technical Report	December 2005

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
8316/25/20	02/124	DCDS(P)	22	3

13. URL on the World Wide Web

http://www.dsto.defence.gov.au/corporate/reports/DSTO-TN-0672.pdf

14. RELEASE AUTHORITY

Chief, Command and Control Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS

Web services; Corporate information management

19. ABSTRACT

This report discusses how Service Oriented Architectures and Web Services were used to make DSTO corporate information available within Microsoft's SharePoint. While the report focuses on exposing of information within SharePoint, the Service Oriented Architecture approach and Web Services technologies are likely to much have wider applicability across the Defence Information Environment.